Ingeniería de Software Clásica y Orientada a Objetos

Sexta Edición, WCB/McGraw-Hill, 2005

Stephen R. Schach
srs@vuse.vanderbilt.edu

EQUIPOS

Índice

- Organización de equipos
- Enfoque del equipo democrático
- Enfoque clásico del equipo con jefe programador
- Más allá de los equipos con jefe programador y democráticos
- Equipos de sincronización-y-estabilización
- Equipos de programación extrema
- Modelo de madurez de la capacidad de los recursos humanos
- Elección de una organización adecuada de equipos

4.1 Organización de Equipos

- Un producto debe estar completo en 3 meses, pero se necesita 1 año-persona de programación
- Solución:
  - Si un programador puede codificar el producto en 1 año, cuatro programadores pueden hacerlo en 3 meses
- ¡Falso!
  - Cuatro programadores probablemente necesitarán cerca de un año
  - La calidad del producto es generalmente inferior

Compartiendo Tareas

- Si un agricultor cosecha un campo de frutillas en 10 días, diez agricultores pueden cosechar el mismo campo de frutillas en 1 día

- Un elefante gesta una cría en 22 meses, pero 22 elefantes no pueden gestar una cría en 1 mes

Compartiendo Tareas (cont.)

- Al contrario que la gestación de los elefantes, sí es posible compartir las tareas de codificación entre los miembros de un equipo
- A diferencia de la cosecha de frutillas, los miembros del equipo deben interactuar de una manera significativa y efectiva

Organización de Equipos de Programación

- Ejemplo:
  - Sheila y Harry codifican dos módulos, digamos `m1` y `m2`
- Qué puede salir mal
  - Tanto Sheila como Harry pueden codificar `m1`, e ignorar `m2`
  - Sheila puede codificar `m1`, Harry puede codificar `m2`. Cuando `m1` llama a `m2`, le pasa 4 parámetros; pero `m2` requiere 5 parámetros
  - O, el orden de los parámetros en `m1` y `m2` puede ser diferente
  - O, el orden puede ser el mismo, pero los tipos de datos pueden ser ligeramente diferentes

Organización de Equipos de Programación (cont)

- Esto no tiene nada en absoluto que ver con la competencia técnica
  - La organización de equipos es una cuestión gerencial

Problemas de Comunicación

- Ejemplo
  - Hay tres canales de comunicación entre los tres programadores que trabajan en un proyecto. La fecha límite se aproxima rápidamente pero el código no está siquiera cerca de completarse

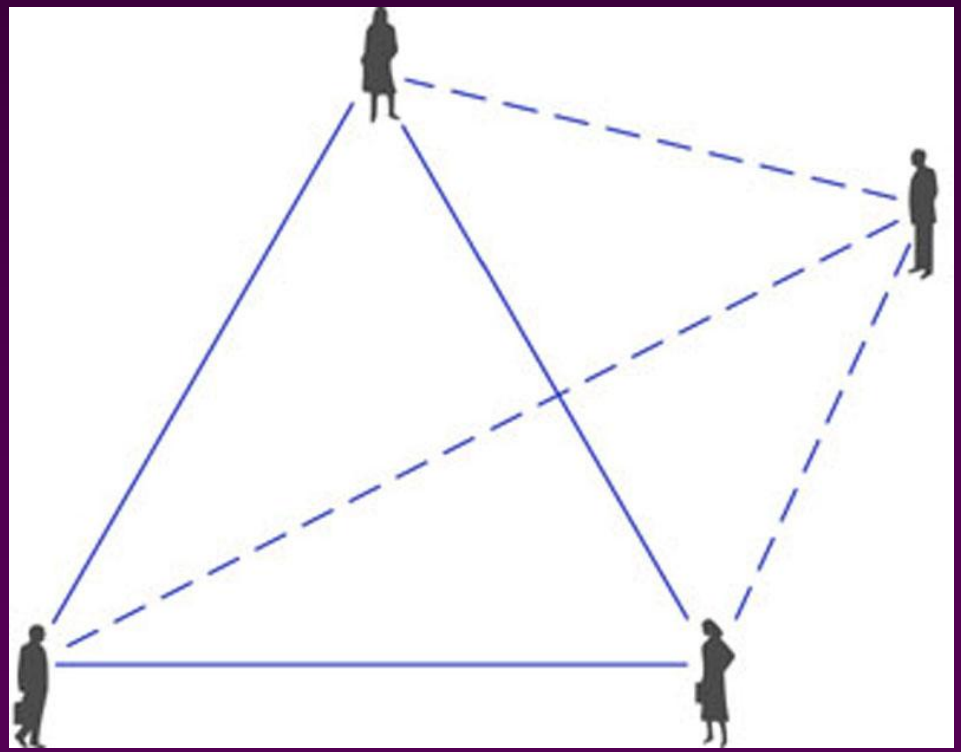

Figura 4.1

- Solución “obvia”:
  - Sumar un cuarto programador al equipo

Problemas de Comunicación (cont.)

- Pero los otros tres tienen que explicar detalladamente
  - Qué se ha hecho
  - Qué está incompleto todavía
- Ley de Brooks
  - Agregar personal adicional de programación a un equipo cuando un producto está atrasado tiene el efecto de hacer que el producto se retrase aún más

Organización de Equipos

- Los equipos se usan a lo largo de todo el proceso de producción del software
  - Pero especialmente durante la implementación
  - Aquí, se presenta la discusión en el contexto de los equipos de programación
- Dos enfoques extremos de la organización de equipos
  - Equipos democráticos (Weinberg, 1971)
  - Equipos con programador jefe (Brooks, 1971; Baker, 1972)

4.2 Enfoque del Equipo Democrático

- Concepto básico subyacente —*programación sin ego*
- Los programadores pueden estar altamente apegados a su código
  - Inclusive llegan a nominar módulos con sus nombres de pila
  - Ven a sus módulos como una extensión de sí mismos

Enfoque del Equipo Democrático (cont.)

- Si un programador ve un módulo como una extensión de su ego, él/ella no va a tratar de encontrar todos los errores en “su” código
  - Si hay un error, se denomina *bug* 
  - La falla podría haber sido evitada si el código hubiese sido mejor protegido contra el “bug”
  - “Shoo-Bug” aerosol spray

Enfoque del Equipo Democrático (cont.)

- Solución Propuesta
- Programación sin ego
  - Reestructurar el ambiente social
  - Reestructurar valores de los programadores
  - Animar a los miembros del equipo a encontrar fallas en el código
  - Una falla debe ser considerada un evento normal y aceptado
  - El equipo como un todo desarrollará un ethos, una identidad de grupo
  - Los módulos “pertenecerán” al equipo como un todo
  - Un grupo de hasta 10 programadores sin ego constituye un *equipo democrático*

Dificultades con el Enfoque del Equipo Democrático

- La gerencia puede tener dificultades
  - Los equipos democráticos son difíciles de introducir en un ambiente no democrático

Fortalezas del Enfoque del Equipo Democrático

- Los equipos democráticos son enormemente productivos
- Trabajan mejor cuando el problema es difícil
- Funcionan bien en un ambiente de investigación
- Problema:
  - Los equipos democráticos deben surgir espontáneamente

4.3 Enfoque Clásico del Equipo con Jefe Programador

- Considere un equipo de 6 personas
  - Quince canales de comunicación de 2-personas
  - El número total de grupos de 2-, 3-, 4-, 5-, y 6-personas es 57
  - Este equipo no puede hacer 6 meses-persona de trabajo en 1 mes

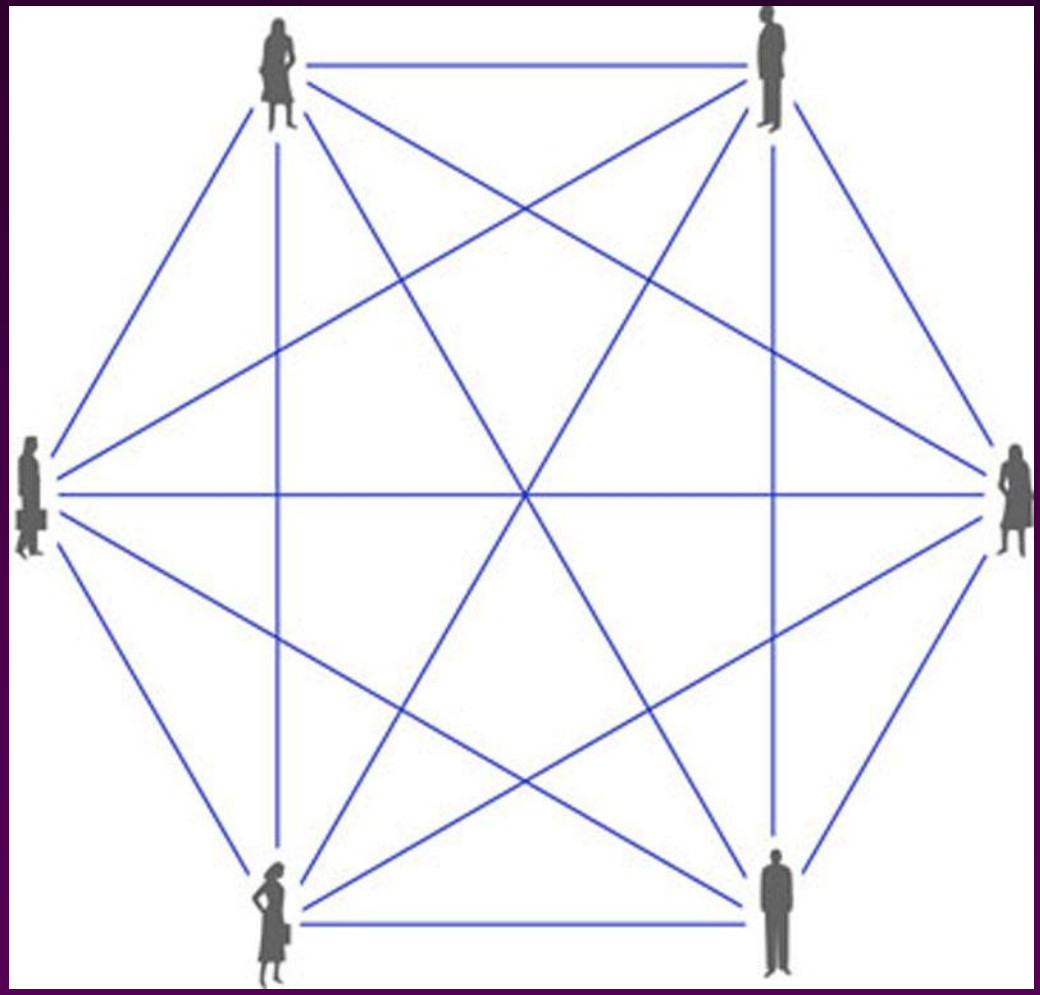

Figura 4.2

Equipo Clásico con Jefe Programador

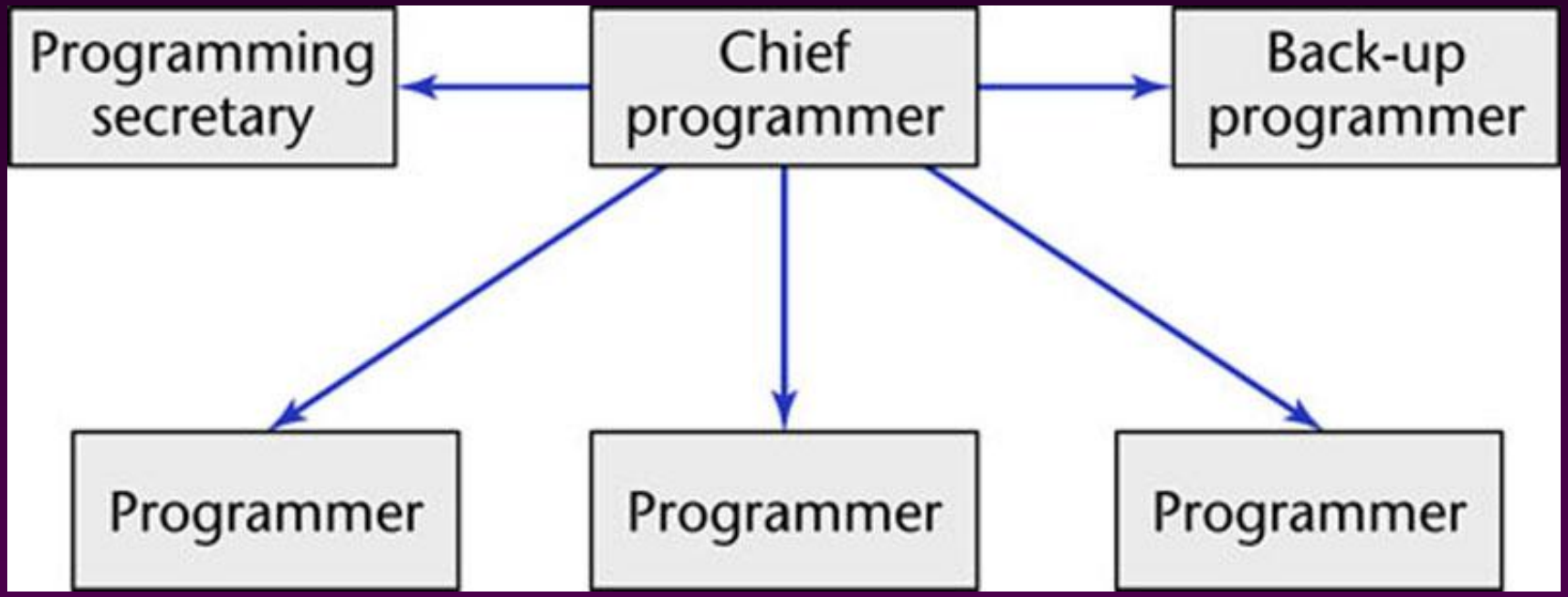

Figura 4.3

- Seis programadores, pero ahora solamente 5 líneas de comunicación

Equipo Clásico con Jefe Programador (cont.)

- La idea básica tras el concepto
  - Analogía: cirujano jefe dirigiendo una operación, asistido por
    - Otros cirujanos
    - Anestesiólogos
    - Enfermeras
    - Otros expertos, tales como cardiólogos, nefrólogos
- Dos aspectos clave
  - Especialización
  - Jerarquía

Equipo Clásico con Jefe Programador (cont.)

- Jefe programador
  - Gerente exitoso *y* programador altamente cualificado
  - Hace el diseño arquitectónico
  - Distribuye la codificación entre los miembros del equipo
  - Escribe las secciones críticas (o complejas) del código
  - Maneja todos los temas de interfaces
  - Revisa el trabajo de otros miembros del equipo
  - Es personalmente responsable por todas las líneas de código

Equipo Clásico con Jefe Programador (cont.)

- Programador de respaldo
  - Necesario solo porque el jefe programador es humano
  - El programador de respaldo debe ser en todo sentido tan competente como el jefe programador, y
  - Debe conocer tanto del proyecto como el jefe programador
  - Hace planificación de casos de prueba de caja negra y otras tareas que son independientes del proceso de diseño

Equipo Clásico con Jefe Programador (cont.)

- Secretario de Programación
  - Un miembro altamente cualificado, bien pagado, y clave del equipo con jefe programador
  - Responsable de mantener la librería de producción de programas (documentación del proyecto), incluyendo:
    - Listados de código fuente
    - JCL
    - Datos de prueba
  - Los programadores pasan su código fuente al secretario, que es el responsable de
    - Convertirlo a lenguaje máquina
    - Compilación, enlazado, carga, ejecución, corrida de casos de prueba (¡1971, recuerden!)

Equipo Clásico con Jefe Programador (cont.)

- Programadores
  - Nada más programan
  - Todos los demás aspectos están a cargo de la secretaría de programación

El Proyecto *New York Times*

- Concepto del equipo con jefe programador
  - Usado por primera vez en 1971
  - Por IBM
  - Para automatizar el banco de datos de recortes de prensa (“morgue“) del *New York Times*
- Jefe programador—F. Terry Baker

El Proyecto *New York Times* (cont.)

- 83,000 líneas de código fuente (LOC) escritas en 22 meses calendario, que representaban 11 años-persona
- Después del primer año, solamente se había escrito el sistema de mantenimiento de archivos (12,000 LOC)
- La mayor parte del código se escribió en los últimos 6 meses
- 21 fallas se detectaron en las primeras 5 semanas de las pruebas de aceptación

El Proyecto *New York Times* (cont.)

- 25 fallas adicionales se detectaron durante el primer año de operación
- Los programadores principales promediaron una falla dectada y 10,000 LOC por año-persona
- El sistema de mantenimiento de archivos, entregado 1 semana después de completarse la codificación, operó 20 meses antes de que ocurra una sola falla
- Casi la mitad de los subprogramas (generalmente 200 a 400 líneas de PL/I) estaban correctos a la primera compilación

El Proyecto *New York Times* (cont.)

- Pero, después de este éxito fantástico, no han vuelto a hacerse reivindicaciones comparables para el concepto de equipo con jefe programador

¿Por Qué Fue Tan Exitoso el Proyecto *NYT*?

- Proyecto prestigioso para IBM
  - Primera prueba real para PL/I (desarrollado por IBM)
  - IBM, con expertos en software magníficos, usó su mejor gente
- Respaldo técnico muy fuerte
  - Desarrolladores del compilador PL/I ayudaban a los programadores
  - Expertos en JCL prestaban asistencia con el lenguaje de control de tareas

¿Por Qué Fue Tan Exitoso el Proyecto *NYT*?

- F. Terry Baker
  - Superprogramador
  - Extraordinario gerente y líder
  - Sus aptitudes, entusiasmo, y personalidad "sostuvieron" el proyecto
- Fortalezas del enfoque del equipo con jefe programador
  - Funciona
  - Numerosos proyectos exitosos han usado variantes del CPT

Poca Practicidad del CPT Clásico

- El jefe programador debe ser un programador altamente capacitado *y* un gerente exitoso
- Hay escasez de programadores altamente cualificados
- Hay escasez de gerentes exitosos
- Es improbable encontrar las cualidades necesarias para ser un programador altamente capacitado en un gerente exitoso, y viceversa

Poca Practicidad del CPT Clásico **(cont.)**

- El *programador de respaldo* debe ser tan bueno como el jefe programador
  - Pero debe usar un asiento posterior (y un salario inferior) esperando a que le pase algo al jefe programador
  - Los mejores programadores, o mejores gerentes no harán eso
- El *secretario de programación* no hace otra cosa que papeleo todo el día
  - Los profesionales de software odian el papeleo
- El CPT clásico es impráctico

4.4 Más Allá de los Equipos CP y Democráticos

- Necesitamos formas de organizar equipos que
  - Hagan uso de las fortalezas de los equipos democráticos y equipos con jefe programador, y
  - Puedan manejar equipos de 20 (o 120) programadores
- Una fortaleza de los equipos democráticos
  - Actitud positiva para encontrar fallas
- Usar CPT en conjunto con revisiones de código o inspecciones

Más Allá de los Equipos CP y Democráticos (cont.)

- Potencial Atascadero
- El jefe programador es personalmente responsable por cada línea de código
  - Debe, por tanto, estar presente en las revisiones
- El jefe programador también es el gerente del equipo
  - Por lo tanto, ¡*no* debe estar presente en las revisiones!

Más Allá de los Equipos CP y Democráticos (cont.)

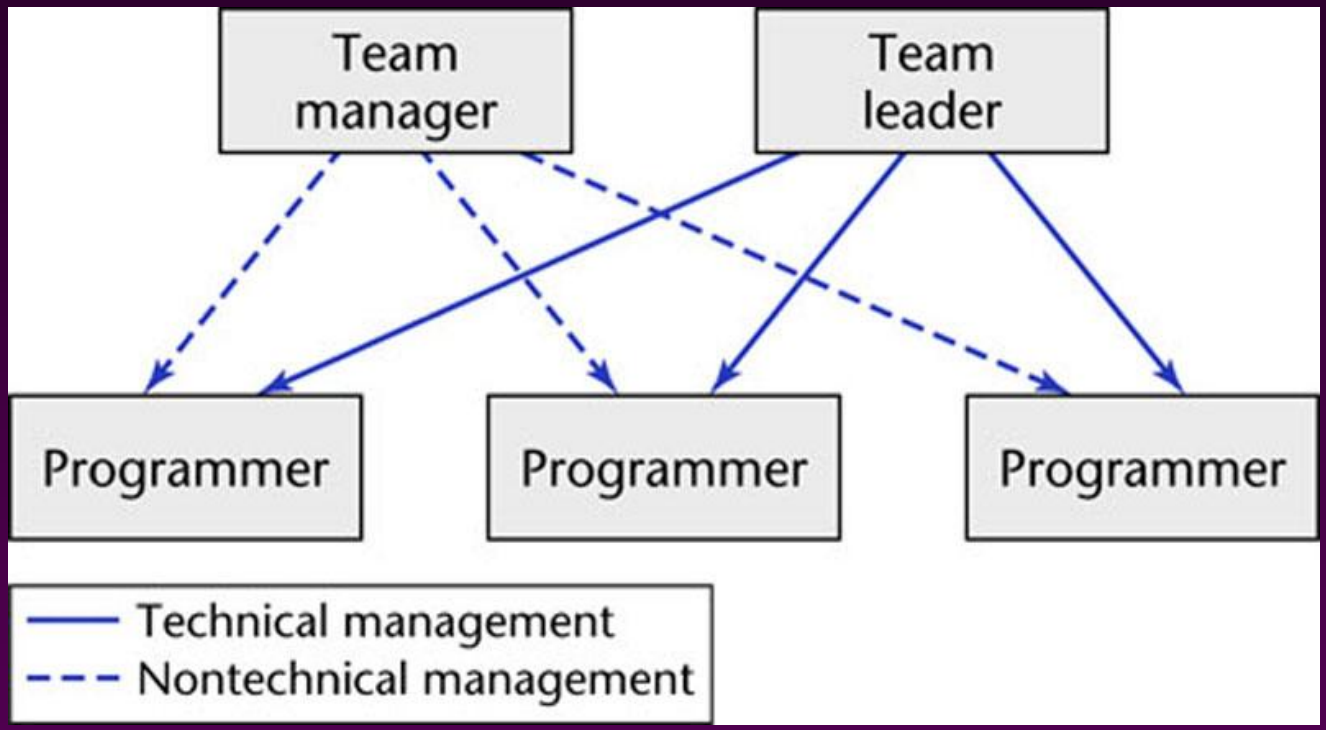

Figura 4.4

- Solución
  - Reducir el rol gerencial del jefe programador

Más Allá de los Equipos CP y Democráticos (cont.)

- Es más fácil encontrar un líder de equipo que un jefe programador
- Cada empleado es responsable ante un gerente únicamente—las líneas de responsabilidad están claramente delimitadas
- El líder del equipo es responsable solamente por la gestión técnica

Más Allá de los Equipos CP y Democráticos (cont.)

- Los asuntos presupuestarios y legales, así como la evaluación de desempeño no son gestionados por el líder del equipo
- El líder del equipo participa en las revisiones—no se permite hacer eso al gerente del equipo
- El gerente del equipo participa en las reuniones regulares del equipo para evaluar la aptitudes técnicas de los miembros del equipo

Proyectos Mayores

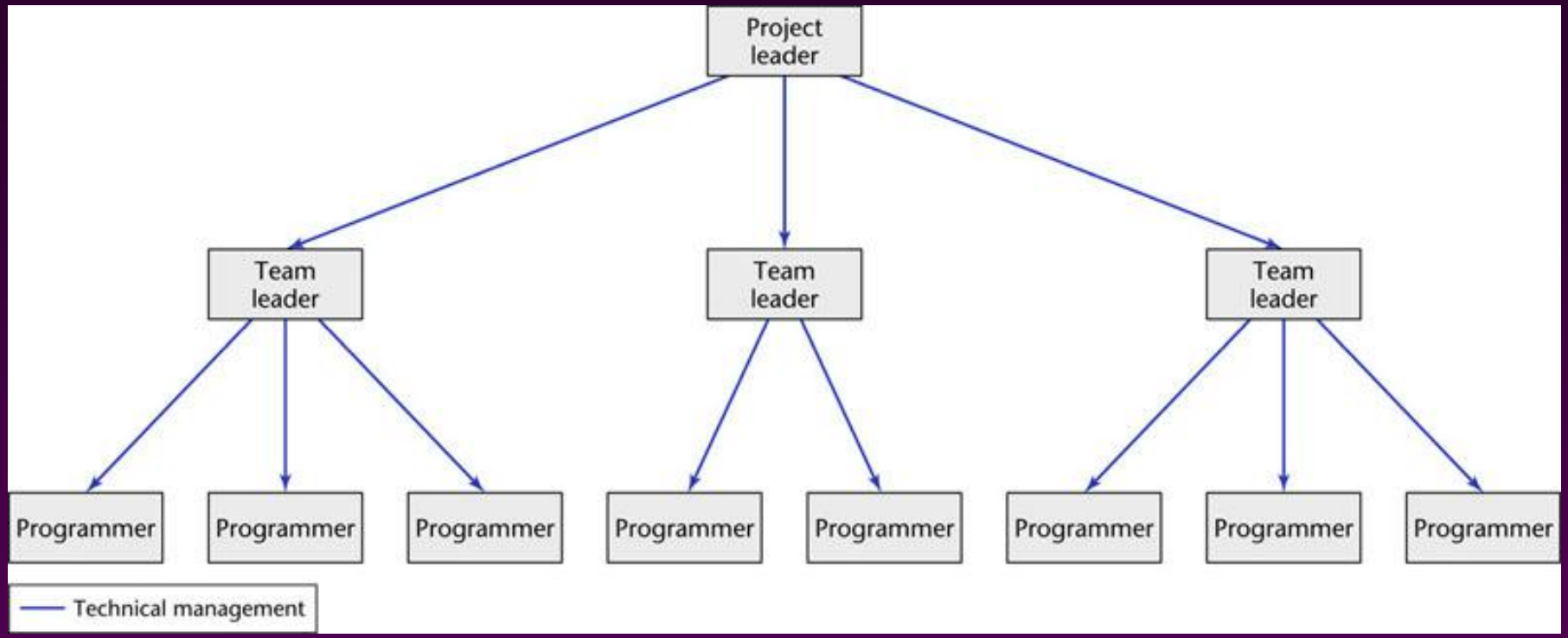

Figura 4.5

- El lado no-técnico es similar
  - Para productos aún mayores, agregar más niveles

Más Allá de los Equipos CP y Democráticos (cont.)

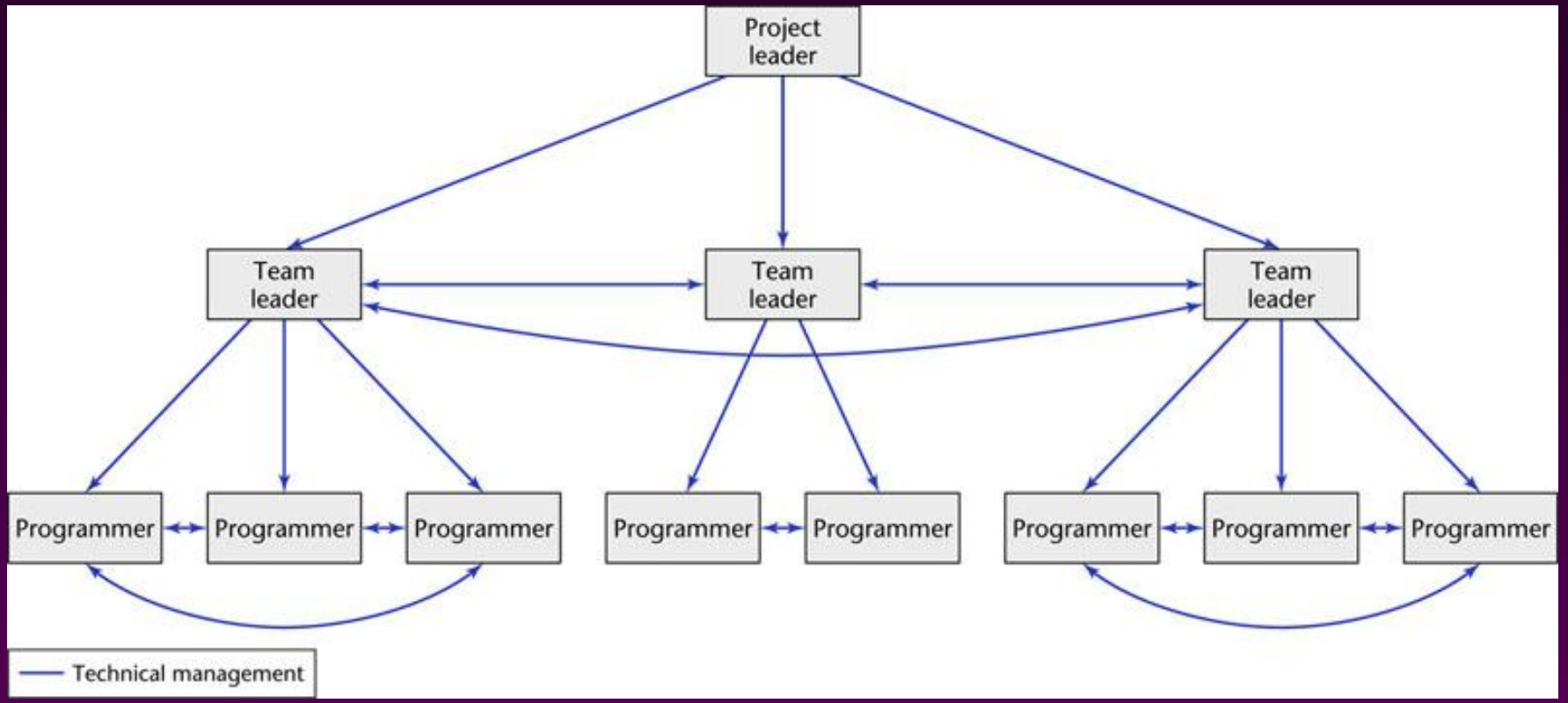

Figura 4.6

- Descentralizar el proceso de toma-de-decisón, cuando sea apropiado
  - Útil cuando el equipo democrático es bueno

4.5 Equipos de Sincronización-y-Estabilización

- Usados por Microsoft
- Los productos comprenden 3 o 4 “construcciones” secuenciales
- Pequeños equipos paralelos
  - 3 a 8 desarrolladores
  - 3 a 8 probadores (trabajan uno-a-uno con los desarrolladores)
  - El equipo recibe la especificación global de la tarea
  - Ellos pueden diseñar la tarea como deseen

Equipos de Sincronización-y-Estabilización (cont.)

- ¿Por qué esto no degenera en un caos inducido por hackers?
  - Paso de sincronización diaria
  - Los individuos integrantes siempre trabajan juntos

Equipos de Sincronización-y-Estabilización (cont.)

- Reglas
  - Los programadores deben apegarse estrictamente a la hora para ingresar el código en la base de datos para la sincronización de ese día
- Analogía
  - Permitir a los niños hacer lo que quieran todo el día…
  - … pero estar en la cama a las 9 P.M.

Equipos de Sincronización-y-Estabilización (cont.)

- ¿Funcionará esto en todas las compañías?
  - Quizás si los profesionales de software son tan buenos como los de Microsoft
- Punto de vista alternativo
  - El modelo de sincronización-y-estabilización es simplemente una manera de permitir a un grupo de hackers desarrollar productos grandes
  - El éxito de Microsoft se debe al excelente marketing más que al software de calidad

4.6 Equipos de Programación Extrema

- Característica de la XP
  - Todo el código es escrito por dos programadores que comparten una computadora
  - “Programación por pares”

Fortalezas de la Programación por Pares

- Los programadores no deberían probar su propio código
  - Un programador detalla los casos de prueba, el otro prueba el código
- Si un programador se va, el otro está suficientemente informado para continuar trabajando con otro par programador
- Un programador inexperto puede aprender de su compañero de equipo más experimentado

Fortalezas de la Programación por Pares

- Los casos de prueba son preparados por un miembro del equipo, y probados por el otro
- El conocimiento no se pierde totalmente si un programador se va
- Un programador novato puede aprender de un colega experimentado
- Las computadoras centralizadas promueven la programación sin ego

4.7 Modelo de Madurez de la Capacidad de las Personas

- Mejores prácticas para gerenciar y desarrollar la fuerza de trabajo de una organización

- Cada nivel de madurez tiene sus propias áreas claves de proceso (KPAs - Key Process Areas)
  - Nivel 2: Selección de personal, comunicación y coordinación, capacitación y desarrollo, ambiente de trabajo, gestión del desempeño, compensación
  - Nivel 5: Mejoramiento continuo de las capacidades, alineación del desempeño organizacional, innovación continua de la fuerza de trabajo

Modelo de Madurez de la Capacidad de las Personas (cont.)

- P–CMM es un marco para mejorar los procesos de una organización para gestionar y desarrollar su fuerza de trabajo
- No se favorece ningún enfoque específico para la organización de equipos

4.8 Eligiendo una Organización de Equipos Apropiada

- No hay una solución al problema de la organización de equipos
- La manera “correcta” depende de
  - El producto
  - El punto de vista de los líderes de la organización
  - La experiencia previa con varias estructuras de equipos

Eligiendo una Organización de Equipos Apropiada (cont.)

- Se ha hecho muy poca investigación sobre la organización de equipos de software
  - En su lugar, la organización de equipos se ha basado en investigación sobre dinámica de grupos en general
- Sin resultados experimentales *relevantes*, es difícil determinar la organización óptima de equipos para un producto específico

Eligiendo una Organización de Equipos Apropiada (cont.)

Team Organization	Strengths	Weaknesses
Democratic teams (Section 4.2)	High-quality code as consequence of positive attitude to finding faults Particularly good with hard problems	Experienced staff resent their code being appraised by beginners Cannot be externally imposed
Classical chief programmer teams (Section 4.3)	Major success of *New York Times* project	Impractical
Modified chief programmer teams (Section 4.3.1)	Many successes	No successes comparable to the *New York Times* project
Modern hierarchical programming teams (Section 4.4)	Team manager/team leader structure obviates need for chief programmer Scales up Supports decentralization when needed	Problems can arise unless areas of responsibility of the team manager and the team leader are clearly delineated
Synchronize-and-stabilize teams (Section 4.5)	Encourages creativity Ensures that a huge number of developers can work toward a common goal	No evidence so far that this method can be utilized outside Microsoft
Extreme programming teams (Section 4.6)	Programmers do not test their own code Knowledge is not lost if one programmer leaves Less-experienced programmers can learn from others Group ownership of code	Still too little evidence regarding efficacy

Figure 4.7